CSC34300 (Spring 2018)
Lab 03: MIPS Assembly Language
(10 points)

IMPORTANT!

Please follow the **submission guidelines** below or your submission will be rejected.

1. You are expected to submit both the lab report and the MIPS assembly source files to Blackboard in a single submission attempt.
2. The MIPS assembly source files should be put into a single ZIP file.
3. Your report and the ZIP file need to be named as follows
    1) The name of your project report should be "FirstName_LastName_Lab_**XX**_Report.pdf". Replace "XX" with the actual project number (2 digits). Note that your project can be a WORD file or a PDF file.
    2) The name of your project file should be "FirstName_LastName_Lab_**XX**_Source.zip". Replace "XX" with the actual project number (2 digits).

In this lab, the students are expected to develop MIPS assembly programs in MARS, an interactive development environment (IDE) for programming in MIPS assembly language.

You will learn from this laboratory the following things:

- The MARS assembler/simulator environment.
- Using MARS.
- Basic MIPS assembly language format.
- The assembler directives *.data, .asciiz, .text, .global.*
- Register notation: `$tn, $an,` and `$vn`.
- Print strings and integers via system calls.
- Exit a program via a system call.
- Procedure calling and recursive procedure calling.
- Use array in your program.
- Construct loop with conditional branch instructions
- The instructions *li, addu, la, move, addiu, beq, beq, j, jal* and *jr*.

MARS has been installed on the class virtual machine. You can also download a copy of MARS from the Internet (http://courses.missouristate.edu/kenvollmar/mars) and install it on your own computer.

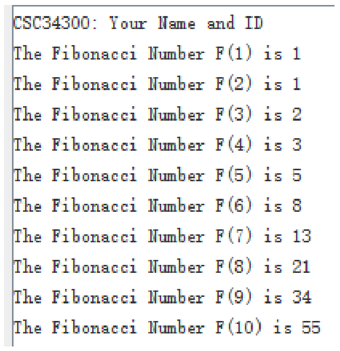
Read the assignment carefully, as you should do for every assignment.

Task 1: Fibonacci Numbers (5 points)

Objectives:

Write a MIPS assembly program to generate Fibonacci numbers. As explained in the lecture, Fibonacci numbers are the numbers in a Fibonacci sequence, which is characterized by the fact that every number after the first two is the sum of the two preceding ones, i.e. F(1)=1, F(2)=1, …, F(n)=F(n-2)+F(n-1).

Expected outputs:



```
CSC34300: Your Name and ID
The Fibonacci Number F(1) is 1
The Fibonacci Number F(2) is 1
The Fibonacci Number F(3) is 2
The Fibonacci Number F(4) is 3
The Fibonacci Number F(5) is 5
The Fibonacci Number F(6) is 8
The Fibonacci Number F(7) is 13
The Fibonacci Number F(8) is 21
The Fibonacci Number F(9) is 34
The Fibonacci Number F(10) is 55
```

Requirements:

1. Your main program should
   a. Display your actual name and student ID number, instead of "Your Name and ID".
   b. Calculate the Fibonacci numbers to F(10).
   c. Call a function (see below) to show the results on the console.
2. Create a function called "print" to
   a. Generate the output using the format "The Fibonacci Number F(**X**) is **Y**", where **X** is the index number and **Y** is the value of the Fibonacci number.
   b. This function therefore should take two input arguments, **X** and **Y**.
   c. Each message should start on a new line.
3. You need to comment **EACH** line of your MIPS assembly codes.

4. List the registers used in your function right before the function implementation.

Note:

1. Your program should follow proper function calling steps.
2. Assume all numbers are signed.
3. Do not use instruction `add` or `addi`. Use `addu` and `addiu` instead.

Deliverables:

1. Your MIPS program, named as `lab1_task1.asm`
   1) Meets all requirements.
   2) No compilation errors from the assembler.
   3) Runs correctly in MARS.
   4) Shows the expected results.
   5) Proper amount of comments (see requirements).
   6) Add the source code into the ZIP file (see submission guidelines)
2. In your report
   1) Include your entire source codes.
   2) Include a screenshot of your program outputs.
   3) Take a look how other addresses (e.g., staring address of a string) is loaded into a register. An address is nothing fancy but a 32-bit value. The code in Task 1 uses "LA", but it actually takes two instructions to load a 32-bit constant into a register. Find out the two instructions for the "LA" operation. And explain how these two instructions get the job done.

Task 2: Fibonacci Number and Array (5 points)

Objectives:

Write a MIPS assembly program to generate Fibonacci numbers, store them in an array, and print out the contents of the array.

The challenge of this task is that you need to determine how many numbers you can compute without an overflow at run-time, i.e. you need to keep computing until you determine that there is an overflow.

Description:

In your program, two steps are expected.

Step 1:

Save the computed Fibonacci numbers in an array. You do not know how many numbers you can compute with 32-bit words. So you will keep computing until you have an overflow.

Use the following code to create an array in the data segment to store the Fibonacci numbers.

```
      FBN:  .space 400
            .align 2
```

The array FBN has 400 bytes. Since each integer has 4 bytes, you can store 100 32-bit integers in the array.

Use the following code to allocate a space to store the index number of the largest Fibonacci number you can compute without an overflow. NUM_FBN is a single 32-bit integer.

```
      NUM_FBN:   .space 4
                 .align 2
```

The pseudocode for this step is as follows.

```
      FBN[0] = 0;
      FBN[1] = 1;
      N = 1;  // The index of the last computed Fibonacci number.
      Do {
            T1 = FBN[N-1];
            T0 = FBN[N];     // load two previous numbers from array
            T0 = T0 + T1;
            If (no overflow) {
                  N = N + 1;
                  FBN[N] = T0;
            }
      } while (no overflow)
      NUM_FBN = N;
```

Step 2:

Write a loop and call the “print” function from Task1 to show the indexes and corresponding numbers stored in `FBN`. The pseudocode is:

```
    For (i = 0; i <= NUM_FBN; i ++) {
          Print FBN[i];
    }
```

Requirements:

1. Your main program should include
   a. Step1: Calculate the Fibonacci numbers until overflow occurs. Save the Fibonacci numbers to an array called `FBN`. Save the last valid index number of the Fibonacci sequence in `NUM_FBN`.
   b. Step2: Call the print function from Task1 to show the numbers stored in the array.
2. You need to comment **EACH** line of your MIPS assembly codes.
3. List the registers used in your function right before the function implementation.

Note:

1. Your program should follow proper function calling steps.
2. Assume all numbers are signed.
3. Do not use instruction `add` or `addi`. Use `addu` and `addiu` instead.

Deliverables:

1. Your MIPS program, named as `lab1_task2.asm`
   1) Meets all requirements.
   2) No compilation errors from the assembler.
   3) Runs correctly in MARS.
   4) Shows the expected results.
   5) Proper amount of comments (see requirements).
   6) Add the source code into the ZIP file (see submission guidelines)
2. In your report
   1) Include your entire source codes.
   2) Explain how you determine if there is an overflow.
   3) Include a screenshot of your program outputs.